DEVELOPMENT TENDENCIES OF PROGRAMMING LANGUAGES

S.J. Mahmudova*

Department of Software Engineering,
Institute of Information Technology of ANAS,
Baku, Azerbaijan.

ABSTRACT

The study presents the information on software engineering (SE) and the knowledge spheres covered by SE. The development stages of programming and the problems encountered in this field are explored. Various programming languages are used to create software for problems. Programming is the process of developing software for a computer. One of the main goals of modern software engineering is to create software products for different problem-oriented areas and to ensure its effectiveness throughout the life cycle. In a broad spectrum, programming means the creation process of software. This includes analyzing and managing the issue, scheduling and designing the program, developing algorithms and data structures, and writing, executing, testing, documenting, configuring, completing and accompanying the program texts. The point is that although the programming languages are powerful and multifunctional, syntax for none of these languages is currently perfect and universal. Hence the programming languages are constantly developing. Broadcasting multi-core processors, cloud programming, portable techniques (digital video camera, music player, mobile phones, GPS navigators, Notebook, İpad, etc.) as well as distributed architecture have created new problems for manufacturers. The information on programming languages are presented, the comparative analysis of those is carried out, and the advantages and drawbacks of each is clarified. Statistical data on the rating of programming languages is presented. Various popular survey agents around the world have compiled a list that rate the programming languages.

KEYWORDS: *Software engineering; programming; programming languages; rating list; algorithm*

1.0 INTRODUCTION

Software engineering is understood as a science that has systematized, arranged methods for the development, testing, maintenance and use of software that is solved by using standards (Matsyashek *et al.*, 2012).

The developed software must meet technical, economic and social requirements. Apparently, development of high-quality software is a process that requires a lot of effort. Processes, tools, technologies and human resources should interact with each other for high quality processing of the project needed. One of the main goals of modern software engineering is

* Corresponding Email: shafagat@gmail.com

to create software products for different problem-oriented areas and to ensure its effectiveness throughout the life cycle.

Software engineering incorporates the following fields of knowledge (Boyko *et al.*, 2007):

a. Fundamentals of computing;
b. Fundamentals of mathematics and engineering;
c. Professional experience (teamwork, communication skills, ethics);
d. Fundamentals of modelling (analysis, work with requirements, specifications);
e. Software design (conceptions and project strategies, design of human-machine interface, project support);
f. Software verification and attestation (testing, user interface evaluation, problem analysis and so on);
g. Basis of software evolution;
h. Software development processes;
i. Software quality;
j. Project management (management concepts, planning and tracking of project implementation, management of staff and configuration).

There are various standards for computers and programming. Many international organizations such as IEEE and ACM have been seriously involved in the development of computers and programming standards. In 2005, the Computer Standards were published (Sommerville, 2011) which include related terms and application of Computer Engineering, Computer Science, Information Systems, Information Technology and Software Engineering.

The standards in these areas are accurate and clearly described. Various programming languages are used to create software for the issues. Programming is a process of software development for computer. The famous Swiss scientist, Wirth Niklaus, who is the author of programming, said: "Programs = Algorithms + Data Structure" (Virt, 2010). The main requirement in programming is the creation of algorithms. The algorithm is the sequence of operations (stages) that are essential to addressing the issue. Generally, the algorithm is a formal writing that identifies the operations needed to resolve a given issue and the sequence in which they are performed. The creation process of algorithms is called algorithmization (Bhargava, 2016). In a broad spectrum, programming means the creation process of the software. This includes analyzing and managing the issue, scheduling and designing the program, developing algorithms and data structures, writing, executing, testing, documenting, configuring, completing and accompanying the program texts.

The first programmer is an English mathematician, Ada Lovalace. As a mathematician, she is famous for writing first program in history and developed her first program for the computing machine invented by Charles Babbage. She also introduced the terms ‘circle’ and ‘cell’. Between 1975 and 1980, a universal programming language was developed and named after Ada Lovelace. The recognition went to the extent of establishing the Ada Prize in 1998 in her memory (Bogazova, 2016).

2.0 PROGRAMMING LANGUAGES

The development trends in the programming languages adhere to some fundamental criteria where it is necessary to provide information about the programming forces of the programming languages. These are ability for future improvement, focus on programming efficiency, ability to considering the complexity of the problem and extending the life cycle of the program. Pratt et al., (2000), gave the review the development trends in programming languages. Seven key approaches used in programming were discussed by Bakett (2013) and shown in Figure 1.

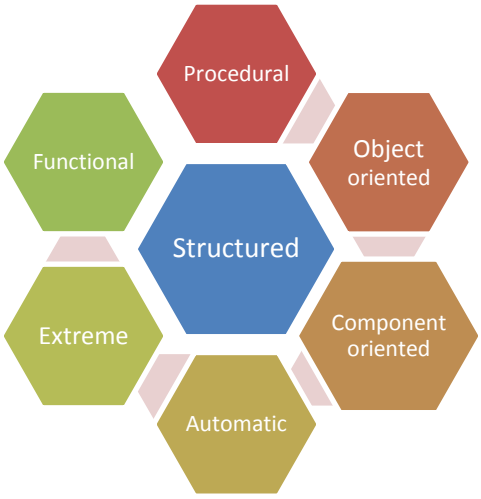

Figure 1. The key approaches used in programming

First programming languages
These languages were created in the 1940 and 1950, in which the program consisted of a "low level" linear sequence of elementary instructions. The advantages of the first programming languages are that it is of high computational efficiency. However, it lacks in terms of significant dependence on the computing environment. Some of the examples of such programming language include machine codes and assembler language.

Procedural (Imperative) programming languages
These languages were created in the 1950 and 1970. Depending on its functional capabilities, it is divided into auxiliary issues, which are also called sub-procedures on the issue. These sub-procedures can be called at any time during the execution of the arbitrary procedural program. The program uses global and local variables. Some of the advantages of this programming language include higher levels of abstraction, low dependence, and broad compatibility. Nonetheless, the main problems highlighted are large labor costs, low code effectiveness. Examples of such programming language are FORTRAN, ALGOL, PL/1, APL, BPL, COBOL, Pascal C and Basic.

Declarative programming languages
These languages were created in 1960. When writing the program, the sequences that need to be implemented are described in simple terms. Simplicity of software verification and

test, sophistication of mathematical formatting, high abstraction were the main advantages of these programming languages. On the other hand, the difficulty of realizing the need for fundamental mathematical knowledge hinders the advancement of the languages. Examples of such programming languages are LISP (Interlisp, Common Lisp, Scheme), SML, Haskell and Prolog.

Functional Programming Languages

Examples of functional programming languages are SML, CaML, Haskell, Miranda, and Hope, which were created in the 1960s. The advantages of the functional programming languages were that they are completely automated management of computer memory, simplicity of reuse of code fragments and covered a wide range of parametric functions (parametric polymorphism). The deficiencies lie on their non-linear structure and relatively low efficiency.

Logical programming languages

The fundamental feature of the program is that the present of a set of rules consisting of logical commands. Created in 1970, the logical programming languages has a high-level abstraction, convenient logical orientation programing and widely applied in expert systems. The limited availability of the issues and the non-linear structure of the software are the main drawbacks that need to overcome. Example of such programming languages are Prolog and Mercury.

Object-oriented programming languages

As the attached to the name, the program describes the objects, aggregates, relationships and methods of interaction between the objects. Advantages of this type of programming languages are that ease of use in modelling, close to subject area, identifiable event orientation, high level of abstraction and easy reuse of images. The main concern of this type of programming languages is its difficulty to verify and test the programs. The examples of such programming languages are the C++, Visual Basic, C #, Eiffel and Oberon, which were developed in 1970 -1980.

Scripting programming languages

These languages were created in 1990, which have the ability to re-use possible scenarios and code. Due to this feature, it has the advantages of close to subject area; high degree of abstraction and high mobility. However, it is difficult to verify and test programs, which share the weakness of object-oriented programming language, as well as exposed to numerous additional effects. VBScript, PowerScript, LotusScript and JavaScript are some examples of such languages.

Parallel programming languages

The examples of the parallel programing languages that created in 1980 were Ada, Modula-2 and Oz. Since its high computational efficiency for large system, these languages were applied in parallel. The language weaknesses lie in its relative low speed in operation and at a much cost to operate.

3.0 MODERN PROGRAMMING LANGUAGES

Currently there are many programming languages and they are used in the development of software products. Such a question arises: Are new programming languages needed? The point is that although the programming languages are powerful and multifunctional, syntax for none of these languages is currently perfect and universal.
Apart this, programming languages are constantly developing. Broadcasting multi-core processors, cloud programming, portable techniques (digital video camera, music player, mobile phones, GPS navigators, Notebook, ipad, etc.). as well as distributed architecture have created new problems for manufacturers. Adding new functionality, paradigms, and templates to existing programming languages that are already available makes them too complicated. In such cases, instead of making new additions to existing programming languages, it is better to start from scratch (Neil, 2013). Thus, 10 advanced programming languages were presented, so solutions for particular problems and specific deficiencies were taken into considerations in each of them. Some of them are already finished, and some are only at the initial stage of their development. It should be noted that, some of them will not gain popularity, but any of them can achieve a revolutionary breakthrough until new languages are replaced with programming.

Some of modern programming languages are shown below:

Dart  . JavaScript is good for adding basic elements of interactivity and creating webpages, but the program code is thousands of lines, which makes it difficult to use the language. That's why this programming language is created as Google Dart's substitute – new language is web-programming language and plays an important role in addressing the reported deficiencies. JavaScript, Dart use similar syntax and keywords in C language. However, there is a significant difference that JavaScript is based on prototypes, while Darts are designated by classes and interfaces. Also Dart allows programmers to add static-type variables, such as C ++ or Java (King, 2011).

Ceylon  . According to Kinğin, Ceylon, "Connects XML with Java". The Ceylon programming language has tried to solve problems in this field (Vikochko, 2011).

Go  **.** Go — is a universal programming language, so it is useful for everyone: From the development of ordinary programs to the development of complex systems, this language can be used. In this sense, it is closer to C and C ++. Go incorporates modern functions, removes unnecessary data from memory, and provides parallelism (Vikochko, 2011).

F#  **.** F# was developed by Microsoft, incorporates functionality and usability in programming. Because F # .NET CLR (Common Language Runtime) is included in first-class languages in virtual machine, it can support libraries and functions as well as other CLR languages (Lazin *et al.,* 2010).

Opa  **.** Opa doesn't replace any language. It is trying to replace all of them by organizing a completely new paradigm in web programming. User interface, database is written in Opa language (Adam 2012).

Fantom **.** Fantom was created for cross-platform portability. The Fantom project is not consisted of only compilers, it separates Java and .Net from API and creates additional level of portability, just as JVM or .Net execute bytecode for CLI (Mulinar, 2009).

Zimbu **.** Zimbu is a unique and specific language for its own character, at the same time also has a large number of features. It uses similar expressions and operators in C, but it owns keywords, data types, and block structures. It manages the memory freely (Talha, 2014).

X10 **.** Parallel programming in the X10 was created based on Model Partitioned global address space (PGAS) model. The codes and data are selected in the blocks and divided into different "spaces" (Lambert *et al.,* 2010).

HAXE . Currently, users can write programs in haXe, then they can compile programs in object code in JavaScript, PHP, Flash / ActionScript, or bytecode NekoVM. Additional modules for C # and Java compilation are being prepared (Benjamin, 2011).

Chapel **.** Chapel is part of the Cray Cascade program, partly funded and used by the American Administration based on a large-scale project on cost-effective calculations in the field of military defense (DARPA). Parallel processing algorithms are used here.

The parallel programming term covers a wide range of areas, as it is related to the calculation of computational systems consisting of several processing devices. Such systems include multi-core processors, machines with common memory multi-core processors, handheld computing packs with distributed memory or hybrid architecture.

Recently, parallel calculations are largely focused on. This is mainly due to two factors. The first factor is related to scientific and technological progress; as new knowledge fields have emerged that require the application of mathematical modeling methods. The second factor is that the models themselves have a considerable complex constructure.

A major advancement in the field of network technology is that they allow the use of local networks of enterprises for parallel calculations (King, 2016).

4.0 THE PROBLEMS OF PROGRAMMING LANGUAGES

Some problems of modern programming languages are shown below:

a. One of the major problems with software developers is the complication of the connection between different systems and their components, due to the increased number of distributed systems. Such problems can be divided into two categories: programming in the small and programming in the large. Local programming problems and their resolution differ according to their complexity and nature.

b. Problems of system variables types.
c. Problems with metadata. Metadata — Describes the structure of the data and its processing methods.
d. Problems occur during implementation of the program.
e. Problems in global programming:
   Naming problem. If geographically diverse producers want to re-use the classes they have developed, there occurs problem of creating different classes with the same names;
   Error handling. Returns are used in a programming language and architecture to provide information about errors, in special cases (Exception). It is necessary to create schemes and architectures allowing to transmit information about errors by maintaining semantic structures of different languages for interactions between such languages, which also causes distress;
   Security. In the design of large distributed systems, most programming languages are used, which in turn causes certain difficulties. That is, security in such cases is not provided;
f. Versioning. Most programmers sometimes encounter problems with the incompatibility of DLL-libraries of the system software, and they realize that the development of the architecture is one of the main problems;
g. Scalability. This is one of the main problems. This is one of the main problems. Distributed systems can work perfectly for hundreds of users on the organization's internal network, but problems can arise when dozens of millions of users are online:
   i. Problems with the size of algorithms;
   ii. Problems with system programming:
   iii. The complexity of modern software;
   iv. Verification and security of programs;
   v. Complex structure of microprocessor;
   vi. Problems with multi-core processors;
   vii. Parallelization of verification and so on.

Various popular survey agents around the world have compiled a rating list of programming languages. The most important of them is TIOBE Index. Figure 2 shows a diagram showing the rating of programming languages for 2016, according to TIOBE Index (Tiobe, 2016).

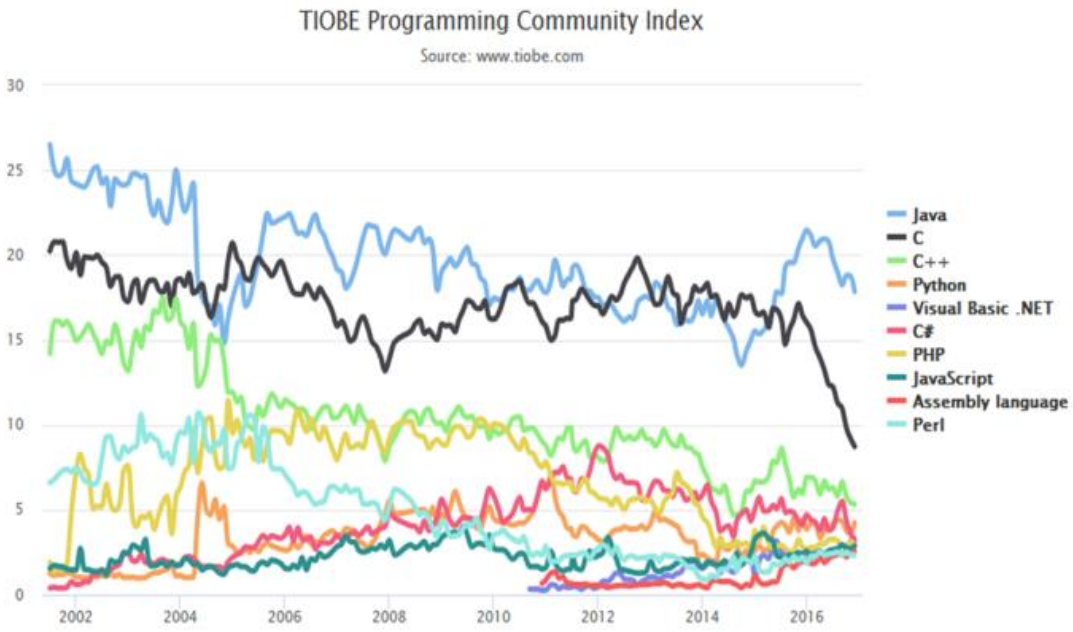

Figure 2. Programming languages rating according to TIOBE Index

5.0 MATHEMATICAL ASPECTS OF PROGRAMMING

Mathematics develops the human mind that developed thinking which able to aids programming. Programming based strongly on the mathematical models or theories. Commonly found mathematical models or theories are Boolean algebra, Linear algebra, Optimization methods and management theory, Calculation methods, Graph theory, Discrete mathematics, Mathematical models, Probability theory, Theory of clusters and Domain theory.

The theory of domain names is considered the branch of mathematics, because it studies the specific types of clusters called partial paramilitary domains. Discrete mathematical structures, graphs, etc. are studied in discrete mathematics. We frequently use whole numbers, graphics and logical expressions in programming. Discrete mathematics can be used in software design, the analysis of algorithms and other practical applications. This is really an excellent tool for the programmer. Apart from these, mathematical programming is one of the main sections of mathematical science, and its object of learning is the complex planning and management of complex systems and processes. Under the term "Intelligent programming", it is intended that the program be designed to compete with the human mind in a convenient way or to assist in solving intellectual problems. From aesthetic point of view, human being is then ensured that he creates everything with his own hands and works under the hands of the master, and from this point of view, the artwork can be viewed as a work of art. Human being is aesthetically pleased when he creates everything himself. From this point of view, programming can be considered as a part of art (Nepeyvoda *et al.*, 2014).

6.0 CONCLUSION

As mentioned, there are many software engineering fields. The most important of them is programming. It is impossible to go to the world market without software. It is possible to create quality software using modern programming languages. As mentioned above, there are shortcomings and problems in this area. Here are some tips for this:

a. Simplify programming of programs;
b. More use of existing components in software development;
c. Preparation of many tests;
d. Ability to work with someone else's program;
e. Choosing the optimal variant for the program to run for a short period of time;
f. Using existing models to extend the life cycle of the program and so on.

The level of language is characterized by the difficulty of solving the task through this language. The more simply the problem solution is written, the more complex operations and concepts are realized, thus, the volume of programs decreases depending on the higher-level language. Modern stages of programming need to be thoroughly studied, and new methods of the software development should be developed based on the knowledge gained. In this regard, manufacturers should use a new methodological system using the algorithmic thinking. It should be noted that the new methods to be used would increase the quality of software.

ACKNOWLEDGEMENT

The author is grateful to Institute of Information Technology of ANAS for the scientific support provided for this research work.

REFERENCES

Adam, K. (2012). *Opa vs. Node.js: What If Things Go Wrong?*. Retrieved from https://www.developer.com/open/opa-vs.-node.js-javascript-framework-comparison.html

Bakett, K. (2013). *Dart in Action.* DMK Press.

Benjamin, D. (2011). *HaXe 2 Beginners Guide.* Packt Publishing.

Bhargava, A. Y. (2016). *Grokking Algorithms.* Manning Publications.

Bogazova, Z. (2016). *Raznyye yazyki programmirovaniya i ikh oblasti primeneniya. Lektsiya v Yandekse.* Retrieved from https://habrahabr.ru/company/yandex/blog/272759/

Boyko, N. I., Zverintsev, M.Ye. (2007). *Rekomendatsii po prepodavaniyu programmnoy inzhenerii i informatiki v universitetakh i dr*. Internet-universitet Informatsionnykh tekhnologiy.

King, G. (2011). *Introduction to Ceylon, Part 10.* Red Hat.

King, S. (2016). *Reyting yazykov programmirovaniya v 2016 godu.* Retrieved from https://habrahabr.ru/company/kingservers/blog/307012/

Lambert, M. S., Miriam T. T., Susan F. M. (2010). *X10 (Programming Language).* Betascript Publishing.

Lazin,Ye., Moiseyev, M., Sorokin, D. (2010). Vvedeniye v F#. *Praktika funktsional'nogo programmirovaniya*, 5:50-105.

Matsyashek, L. A., Liong, B. L. (2012*). Prakticheskaya programmnaya inzheneriya na osnove uchebnogo primera*. Izdatel'stvo Binom: Laboratoriya znaniy.

Mulinar, B. (2009). *Novyy yazyk programmirovaniya*. Retrieved from http://eao197.blogspot.com/2009/10/compprogflame-zimbu.html

Neil, M. (2013). *10 Yazykov programmirovaniya, kotoryye mogut perevernut mir*. Retrieved from https://www.kv.by/content/325498-10-yazykov-programmirovaniya-kotorye-mogut-perevernut-mir-it.

Nepeyvoda, N. N., Grigorevskiy, I. N. Lilitko, Ye. P. (2014). *Matematicheskiye osnovy programmirovaniya.* Retrieved from http://psta.psiras.ru/read/psta2014_4_105-121.pdf

Pratt, T. W., Zelkovitz, M. V. (2000). *Programming languages, design and implementation.* Prentice Hall.

Sommerville, I. (2011). *Software Engineering.* Inc., publishing as Addison-Wesley.

Talha, K. (2014). *OpenFL & Haxe, a Bumpy Start*. Retrieved from https://en.wikipedia.org/wiki/Haxe

Tiobe. (2016). *TIOBE Index for 2016.* Retrieved from http://www.tiobe.com/tiobe-index/

Vikochko, D. (2011). *Yazyk programmirovaniya go*. Developer Works.

Virt, N. (2010). *Algoritmy i struktury dannykh.* M.: DMK Press.